laboration 3

gean piere ventura cruz

2022-03-07

Uppgift 1

Vi kan definera tillståndsbyten som en poisson-process. Sedan har vi en så kallad M/M/2-kö process som då sätts igång när datorn börjar exekvera jobb och jobb som lagras i buffert (2 st). Detta system kan beskrivas med hjälp av en så kallad balansekvation som säger att summan av inflödet och utflödet är lika med 0. Dessa kan defineras som matriserna P respektive Q. Matrisen Q:s element är de intensitetsparametrar och kan ställas upp på följande sätt:

$$Q = \begin{pmatrix} -\lambda & \lambda & 0 & 0 \\ \mu & -(\lambda+\mu) & \lambda & 0 \\ 0 & \mu & -(\lambda+\mu) & \lambda \\ 0 & 0 & \mu & -\mu \end{pmatrix}$$

och för matrisen P får vi:

$$p = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{\mu}{\lambda+\mu} & 0 & \frac{\lambda}{\mu+\lambda} & 0 \\ 0 & \frac{\mu}{\lambda+\mu} & 0 & \frac{\lambda}{\mu+\lambda} \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Kommande jobb som görs är då enligt förutsättningarna poissonfördelade. Samma sak gäller för tiden för att dessa ska exekveras, med intensitet mu. Detta i sin tur medför att fördelningen för tiden till nästa tillståndsbyte är exponentielfördelad med intensitet mu + lambda. Tiden till nästa tillståndsbyte (TB) är då iid enligt ~ Exp(my + lambda). Alltså P(TB≤t)=1-exp^-(mu+lambda)t. För tillstånden 0 och 3 gäller det att mu = lambda = 0. Sannolikheten att processen går upp eller ned ett steg är sannolikheten för en nyankomst minus sannolikheten för samma tillstånd att den sjunker. Vi kan skriva sannolikheten att går från tillstånd 0 till tillstånd 1 som P(0->1) = 1-exp^(-lambda*t), och om ifall en ökning av tillstånd skulle ske från 1 till 2 kan vi skriva detta som P(1->2) = P(0->1).

Uppgift 2

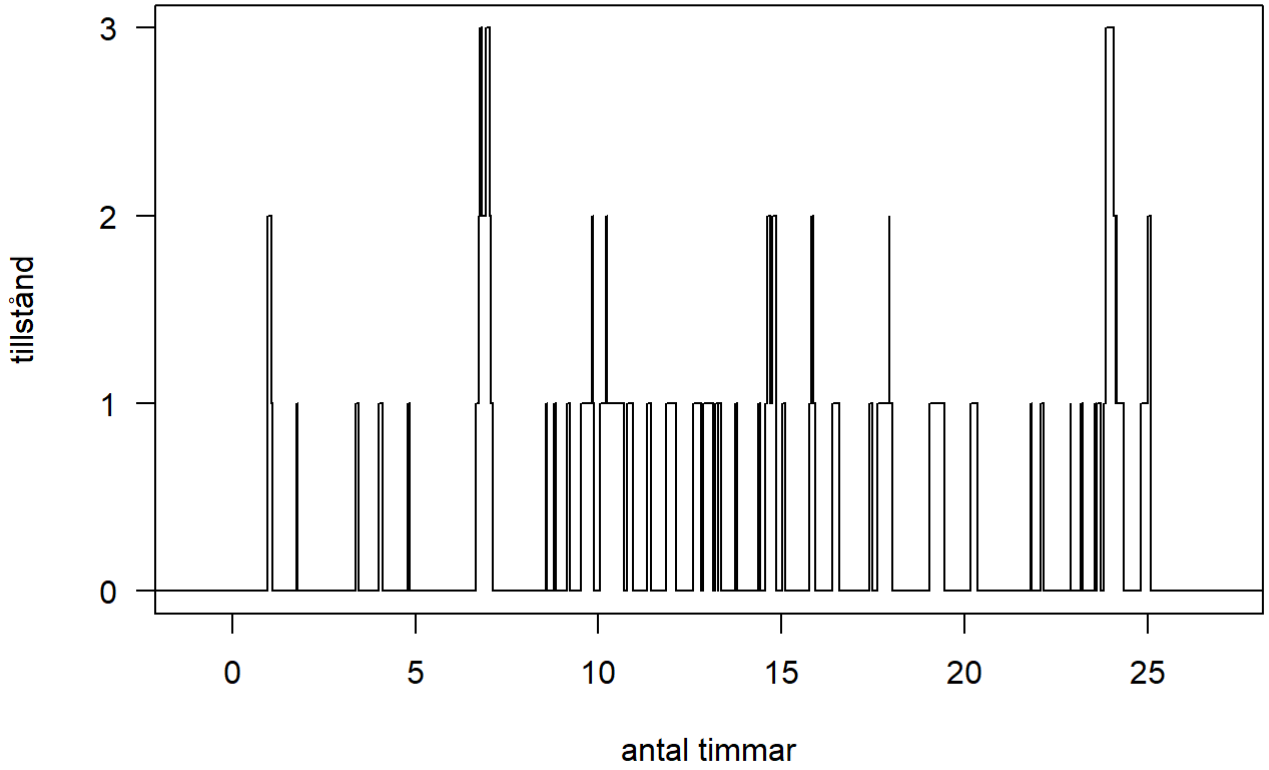
```
bd_process = function(lambda, mu, initial_state = 0, steps = 100) {
  #en simulering som avbryts efter n-antal steg och stannar i en av de 4 tillstånden i S.
  time_now = 0
  state_now = initial_state
# Dessa vektorer ska byggas på i loopen nedan
  time = 0
  state = initial_state
  for (i in 1:steps) {
# Ett villkor som säger att om vi befinner oss i tillstånd 3 så sätts lambda till 0.
# Alltså inga fler jobb kan tas emot annars ta vi emot ett nytt jobb med intensitet lambda.
    if (state_now == 3) {
        lambda_now = 0
        } else {
          lambda_now = lambda
        }
# När datorn är ledig finns det tillgång att ta emot nya jobb. Då sätts parametern mu
# lika med 0 då inga jobb tas hand om under tiden. Annars avslutas jobbet med intensitet mu.
    if (state_now == 0) {
        mu_now = 0
        } else {
          mu_now = mu
    }
# tid till nästa övergång. denna tid räknas ut genom att ange ett slumpmässigt som tas
# som argument till fördelning exp(lambda + mu)
  time_to_transition = -(1/(mu_now + lambda_now))*log(runif(1))
# Om sannolikheten för ett avslutat jobb är större än sannolikheten för att starta ett nytt
# jobb så går det till ett lägre tillstånd -1. Annars ett högre tillstånd +1.
  if (((mu_now + lambda_now)*runif(1)) < mu_now) {
          state_now = state_now - 1
     } else {
          state_now = state_now + 1
            }
    time_now = time_now + time_to_transition # en sammansättning av tider (vektor) som
                                             # sker mellan hoppen av tillstånd och adderas
                                             # med tiden till nästa övergång
    time = c(time, time_now) #en vektor som innehåller tidspunkt för den den senaste tillstån
dshopp
    state = c(state, state_now) # vad innehåller vektorn state? Vårt nuvarande tillståndsrum
                                #(0,...,3) och tillstånden som redan genomlöpts (initial_s
tate)
  }
# Returnera en lista med de två vektorerna tid och state
  list(tid = time, state = state, steps=steps)
}
```

Uppgift 3

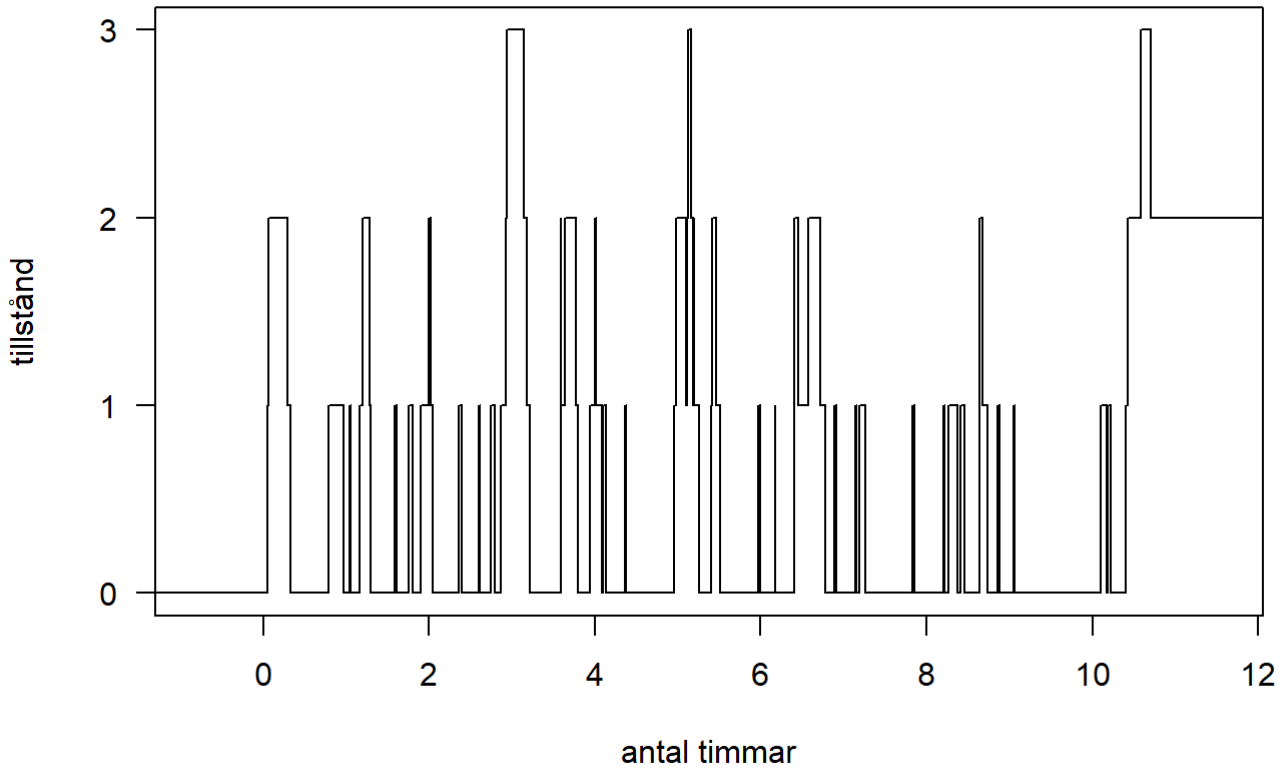
Vi ska rita upp ett simulerat trappstegsdiagram för vart och ett av förslagen 1–3.

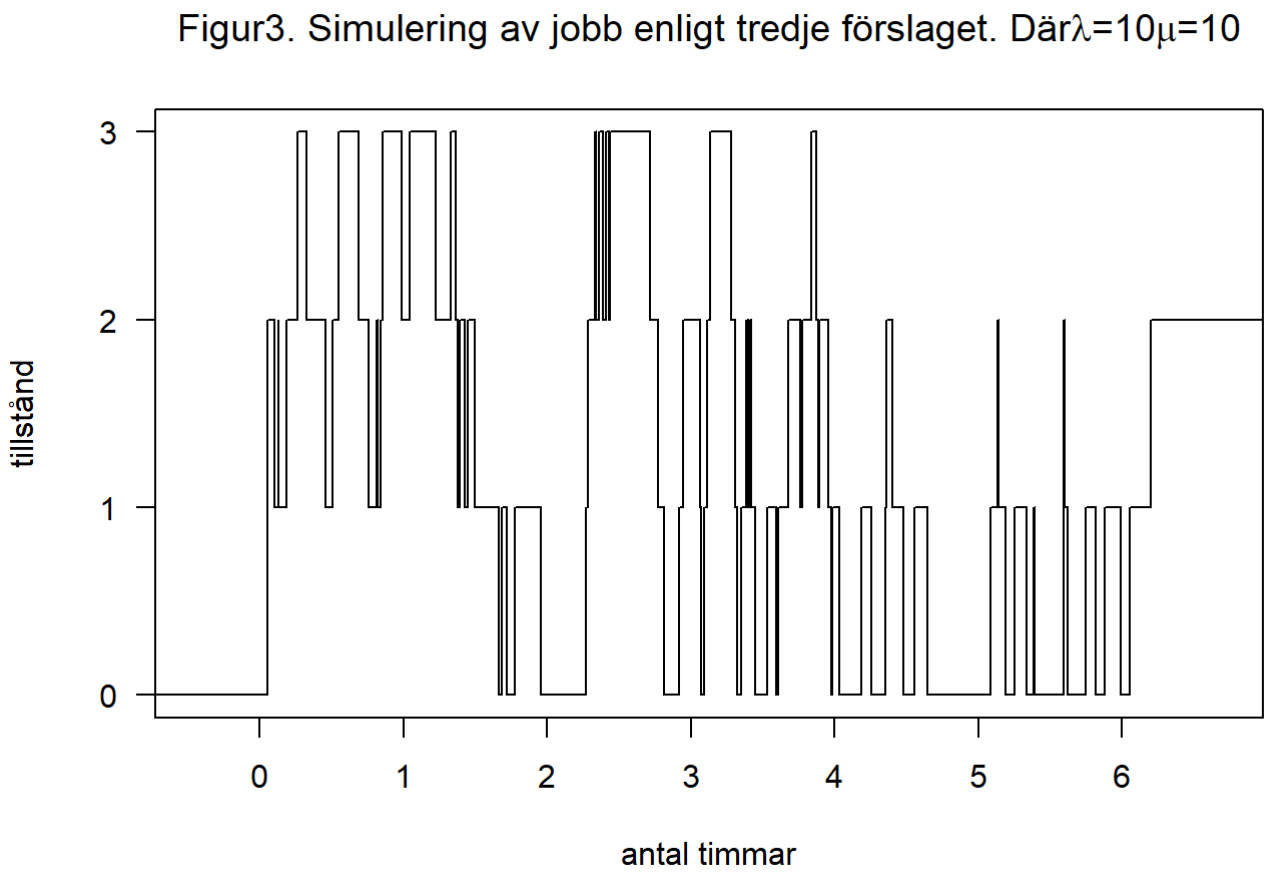
```
{set.seed(19960618)
#förslag 1 med parameterna mu = 2, lambda=10, n/steps=100
forslag1 = bd_process(2, 10, 0, 100)

time1 = forslag1$tid
state1 = forslag1$state
steps1 = forslag1$steps

plot(stepfun(time1[-1], state1),
     do.points = FALSE,
     xlab = "antal timmar",
     ylab = "tillstånd",

     main = expression(paste("Figur1. Simulering av jobb enligt första förslaget. Där", lambd
a, "=2", mu, "=10")),
     yaxt = "n")
     axis(2, at = c(0, 1, 2, 3), las = 2)

#förslag 2 med parameterna mu = 6, lambda=10, n/steps=100
forslag2 = bd_process(6, 10, 0, 100)

time2 = forslag2$tid
state2 = forslag2$state
steps2 = forslag2$steps

plot(stepfun(time2[-1], state2),
     do.points = FALSE,
     xlab = "antal timmar",
     ylab = "tillstånd",
     main = expression(paste("Figur2. Simulering av jobb enligt andra förslaget. Där", lambd
a, "=6", mu, "=10")),
     yaxt = "n")
     axis(2, at = c(0, 1, 2, 3), las = 2)

#förslag 3 med parameterna mu = 10, lambda = 10, n/steps=100
forslag3 = bd_process(10, 10, 0, 100)

time3 = forslag3$tid
state3 = forslag3$state
steps3 = forslag3$steps

plot(stepfun(time3[-1], state3),
     do.points = FALSE,
     xlab = "antal timmar",
     ylab = "tillstånd",
     main = expression(paste("Figur3. Simulering av jobb enligt tredje förslaget. Där", lambd
a, "=10", mu, "=10")),
     yaxt = "n")
     axis(2, at = c(0, 1, 2, 3), las = 2)
}
```

Figur1. Simulering av jobb enligt första förslaget. Därλ=2μ=10



Figur2. Simulering av jobb enligt andra förslaget. Därλ=6μ=10





Figurerna visar vilka som ska ha tillträde till datorn. Denna spridning verkar hamna i en högre tillstånd ju fler jobb som görs per timme. Alltså “vem som helst har tillträde”.

Uppgift 4

```
#simulation av 500 steg för förslag 1:

forslag_1 = bd_process(lambda = 2, mu = 10,steps=500)
time_1 = forslag_1$tid
state_1 = forslag_1$state

#simulation av 500 steg för förslag 2:

forslag_2 = bd_process(lambda = 6, mu = 10, steps = 500)
time_2 = forslag_2$tid
state_2 = forslag_2$state

#simulation av 500 steg för förslag 3:

forslag_3 = bd_process(lambda = 10, mu = 10, steps = 500)
time_3 = forslag_3$tid
state_3 = forslag_3$state


tid = data.frame(tid_h = c(floor(time_1[length(time_1)]), trunc(time_2[length(time_2)]), roun
d(time_3[length(time_3)], digits=0)))
rownames(tid) = c( "Förslag 1", "Förslag 2", "Förslag 3")
knitr::kable(tid, digits = 3)
```

	tid_h
Förslag 1	123
Förslag 2	48
Förslag 3	33

tabell 1 visar hur lång tid det tog innan systemet hade ändrat tillstånd 500 gånger, för respektive förslag.

Vi ser i tabellen att tiden i antal simuleringar ökar då aktiviteten minskar alltså då mindre färre jobb per timme exekveras och tvärtom.

Uppgift 5

```
forslag1 = bd_process(2, 10, 0, 1000)
forslag2 = bd_process(6, 10, 0, 1000)
forslag3 = bd_process(10, 10, 0, 1000)

proportion_in_state = function(s, bdp){
  räkning = 0
  tid = bdp$time
  läge = bdp$state
  steg = bdp$steps
  for (i in 1:steg){ #itererar över alla steg
    if (s == läge[i]) #checkar om steg i är lika med den givna tillståndet
        räkning = räkning + (tid[i+1]-tid[i])
  }
  tids_proportion = räkning/tid[steg]#divideras för att få andelen
  return(tids_proportion)
}
```

```
tids_prop=matrix(c(0.9082885, 0.07928075, 0.006131374, 0.006299343,
              0.5018268, 0.23489090, 0.149956705, 0.113325583,
              0.2496250, 0.23610256, 0.207728505, 0.306543916),
            byrow = TRUE,
            dimnames = list(c("forslag1","forslag2","forslag3"), c("läge0","läge1","läge2","l
äge3")),
            nrow = 3,
            ncol = 4)
knitr::kable(tids_prop, digits = 2)
```

	läge0	läge1	läge2	läge3
forslag1	0.91	0.08	0.01	0.01
forslag2	0.50	0.23	0.15	0.11
forslag3	0.25	0.24	0.21	0.31

```
#t(tids_prop)
```

Matrisen visar för vart och ett av förslagen hur står del av tiden dessa tillbringats i respektive tillstånd. Vi kallar matrisen tids_prop för att visa förhållandet mellan andel av totala tiden i vart och ett av förslagen respektive förslag.

```
tabell= data.frame(Forslag = c(" förslag 1", " förslag 2", "förslag 3" ) )
tabell= cbind(tabell, rbind(tids_prop[1,], tids_prop[2,], tids_prop[3,]))

names(tabell)[-1]= paste0("tillstånd ", 0:3)
knitr::kable(tabell, digits=4)
```

Forslag	tillstånd 0	tillstånd 1	tillstånd 2	tillstånd 3
förslag 1	0.9083	0.0793	0.0061	0.0063
förslag 2	0.5018	0.2349	0.1500	0.1133
förslag 3	0.2496	0.2361	0.2077	0.3065

tabell 2 visar andelen uppehållstid i de 4 tilllstånden.

Förslag 1 talar om för oss att ett underutnyttjande av systemet sker då systemet har full beläggning av ca 0.6% av tiden jobben körs igång. Förslag 2 talar om för oss å andra sidan att här sker det en ca 10% av drifttiden. Förslag 3 talar om för oss att systemet är fullt nyttjat av ungeför 25% av drifttiden. Då ger systemet användaren ett felmeddelande. Då ses inte denna förslag som ett bra förslag då kravet var att “högst 5% av alla jobb som skickas till datorn ska behöva råka ut för detta”. Som slutsats kan förslag 2 ändå ses vara lämpligast.

Uppgift 6

Vi ska ange en formel för den stationära fördelningen, Π, som funktion av kvoten $\rho = \frac{\lambda}{\mu}$. Om $\lambda \neq \mu$ så får vi:

$$\Pi_n = \frac{\rho^n(1-\rho)}{(1-\rho^{N+1})}$$

med begränsning N.

I fallet där $\lambda = \mu$ får vi $\frac{1}{N+1}$

Under förutsättningen att processen är minneslös fås följande balansekvation:

$$\mu_1 P_1 = \lambda_0 P_0$$
$$(\mu_1 + \lambda_1)P_1 = \lambda_0 P_0 + \mu_2 P_2$$
$$\vdots$$
$$(\mu_n + \lambda_n)P_n = \lambda_{n-1} P_{n-1} + \mu_{n+1} P_{n+1}$$

Vi kan nu skriva om denna ekvation som:

$$P_1 = \frac{\lambda_0}{\mu_1} P_0$$
$$\vdots$$
$$P_n = \frac{\lambda_0 \cdot \cdot \lambda_{n-1}}{\mu_1 \cdot \cdot \mu_n} P_0 = \rho_n P_0$$

Där n≥1 och rho är trafiksintensiteten för n steg. Eftersom vår process är stationär kan vi förenkla och vi får att

lambda_n = lambda och mu_n=mu. Vi vet också följande att:

$$\sum_{n=0}^{\infty} P_{0n} = P_0 + \rho_1 P_0 + \rho_2 P_0 \ldots = P_0(1 + \sum_{n=0}^{\infty} \rho_n) = 1 \Rightarrow$$

$$\Rightarrow P_0 = \frac{1}{1 + \sum_{n=0}^{\infty} \rho_n} = \frac{1}{1 + \sum_{n=0}^{\infty} \frac{\lambda}{\mu}}$$

I vårt fall körs processen till då n = 0-3.

Vi ställer upp två funktioner som räknar ut i respektive fall:

```
N = 3 #antal tillstånd

#vi skapar en tabell förvarar all data

tabell = matrix(nrow = 3,
                ncol = 4,
                byrow = T)

#förslag 1

rho = 2/10
tabell[1,1:4] = rho^(0:3)*(1-rho)/(1-rho^(N+1))
tabell[1,1:4]
```

```
## [1] 0.801282051 0.160256410 0.032051282 0.006410256
```

```
#förslag 2

rho = 6/10
tabell[2,1:4] = rho^(0:3)*(1-rho)/(1-rho^(N+1))
tabell[2,1:4]
```

```
## [1] 0.45955882 0.27573529 0.16544118 0.09926471
```

För förslag 1 som har mu = 2 och lambda 10 får vi rho = 1/5=0,2:

$$P_0 = 1 - \rho = 0.801$$
$$P_1 = \rho P_0 = 0,2 \cdot 0,8 = 0,160$$
$$P_2 = \rho^2 P_0 = 0,2^2 \cdot 0,8 = 0,0320$$
$$P_3 = \rho^3 P_0 = 0.2^3 \cdot 0.8 = 0.0064$$

Vi ser risken för full beläggningen av systemet är 0.64%.

För förslag 2 som har mu = 6 och lambda 10 får vi rho = 3/5=0,6:

$$P_0 = 1 - \rho = 0.459$$
$$P_1 = \rho P_0 = 0,6 \cdot 0,4 = 0,275$$
$$P_2 = \rho^2 P_0 = 0,6^2 \cdot 0,4 = 0,165$$
$$P_3 = \rho^3 P_0 = 0.6^3 \cdot 0.4 = 0.099$$

Vi ser risken för full beläggningen av systemet är 9.9%.

För förslag 3 som har mu = 10 och lambda 10 får vi rho = 1:

$$P_3 = P_2 = P_1 = P_0 = \frac{\rho^3}{1+3\rho} = 0.25$$

Vi ser risken för full beläggningen av systemet är 25%.

```
P = matrix(c(0.801, 0.160, 0.032, 0.0064,
               0.459, 0.275, 0.165, 0.099,
               0.25, 0.25, 0.25, 0.25),ncol=4,byrow=TRUE)
colnames(P) = c("tillstånd0","tillstånd1","tillstånd2","tillstånd3")
rownames(P) = c("forslag1","forslag2","forslag3")
P = as.table(P)
knitr::kable(P, digits=4)
```

	tillstånd0	tillstånd1	tillstånd2	tillstånd3
forslag1	0.801	0.160	0.032	0.0064
forslag2	0.459	0.275	0.165	0.0990
forslag3	0.250	0.250	0.250	0.2500

tabell3 visar beräkning av stationär fördelning. Andelen uppehållstid i de 4 tillstånden respektive förslag

Som vi ser i tabell 3 så är risken för full beläggning av systemet ungefär 10%. Detta överskrider kravet som är högst 5% risk. Då maximalt risk ska inte överskrida 5%, bör då förslag 1 införas. Förslag 1 i tabell 3 har 0.64% för risk för full beläggning av systemet, alltså att tillkommande arbete avvisas.

Att avvisa alla elever i de datorer som krävs av systemet är dock inte ett kostnadseffektivt sätt att använda systemet. Institutioner kan ha rykte om bristande kompetens och bristande förmåga att hantera effektiv användning av resurser. Därför bör styrelsen överväga att få tillgång till och använda systemet enligt det förslag 2 enligt prioritetsordningen mellan de två användarna. I det här fallet innebär förslaget som kallas förslag 4 att endast arbetstagare tilldelas prioritet 1, så de är alltid ledande i arbetskön. Studenter ska ha rätt att använda den här resursen, men prioriteten är prio-2. Studenten ska kunna acceptera den personal som leder kön. Arbetsprioriteten kan lösas tekniskt genom buffersystemet. Prioritet 1-arbete utförs alltid innan, och vid behov prio 2-arbete “skjuts tillbaka” eller “skjuts” från kön.